МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Алгоритмы и структуры данных»

Тема: Случайные БДП с рандомизацией. Текущий контроль

Студент гр. 8304 _____ Ивченко А.А.

Преподаватель _____ Фирсов

Санкт-Петербург

2019

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Ивченко А.А.

Группа 8304

Тема работы: Случайные БДП с рандомизацией. Текущий контроль

Исходные данные:

Случайные БДП с рандомизацией. Текущий контроль.

Содержание пояснительной записки:

- Введение
- Описание алгоритма
- Описание функций и сд
- Тестирование
- Заключение
- Список используемых источников
- Приложение А

Предполагаемый объем пояснительной записки:

Не менее 34 страниц.

Дата выдачи задания: 14.10.2019

Дата сдачи реферата:

Дата защиты реферата:

Студент _____ Ивченко А.А.

Преподаватель _____ Фирсов К.В.

АННОТАЦИЯ

В данном курсовом проекте была разработана система текущего контроля среди студентов по теме «Случайные бинарные деревья поиска с рандомизацией». Программа генерирует задания, согласно которым студенты, проходящие ТК, должны выполнить вставку в рандомизированное БДП. На основании решенных тестов выводится рейтинговая оценка знаний студентов. Здесь приведено полное описание работы алгоритм и тестирование программы. Исходный код на языке C++ содержится в Приложении А.

SUMMARY

In this project a system of ongoing monitoring on the topic "Random binary search trees with randomization" was developed. Program generates tasks, under which students must to do an insert into a randomized BST. Based on the solved tests, a rating assessment of student knowledge is derived. Here is a complete description of the algorithm and testing the program. C ++ source code is provided in Appendix A.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является написание такой программы, которая позволит дифференцировать знания студентов по теме «Случайные БДП с рандомизацией».

1. ОПИСАНИЕ АЛГОРИТМА.

Для реализации этой задумки, была реализована система Текущего Контроля.

1.1. Заголовочные файлы

К рабочему файлу main.cpp был подключен хедер btree.h, функции которые подробнее рассматриваться в следующем разделе. Заголовчный файл содержит структурную реализацию бинарного дерева поиска с рандомизацией, а также все необходимые методы для работы с ним.

1.2. main.cpp

В файле main.cpp представлен алгоритм программы, который работает следующим образом. В функцию tasksGenerating подаются отрытые файлы для записи заданий и ответов(ofstream) и рандомизированное БДП. Функция tasksGenerating обеспечивает генерацию билетов с заданиями. Задания сформулированы следующим образом:

Задание: вставка с рандомизацией узла с весом n

- Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.
- Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

Задания создаются со случайными параметрами, которые определяет функция генерации случайных чисел rand.

После генерации всех вариантов с заданиями и ответов к ним, пользователю предлагается перейти в файл с именным названием и написать ответы к задачам, соответствующим своему варианту. При подтверждении готовности, программа сверяет ответы и выводит результат.

2. ОПИСАНИЕ ФУНКЦИЙ И СД.

2.1. Общие сведения

Рандомизированное бинарное дерево поиска (англ. *Randomized binary search tree, RBST*) — структура данных, реализующая бинарное дерево поиска.

Как известно, можно подобрать такую последовательность операций с бинарным деревом поиска в наивной реализации, что его глубина будет пропорциональна количеству ключей, а следовательно операции будут выполняться за O(n)O(n). Поэтому, если поддерживать инвариант "случайности" в дереве, то можно добиться того, что математическое ожидание глубины дерева будет небольшим. Дадим рекурсивное определение рандомизированного бинарного дерева поиска (RBST).

Определение. Пусть TT — бинарное дерево поиска. Тогда

1. Если TT пусто, то оно является **рандомизированным бинарным деревом поиска**.
2. Если TT непусто (содержит nn вершин, n>0n>0), то TT — **рандомизированное бинарное дерево поиска** тогда и только тогда, когда его левое и правое поддеревья (LL и RR) оба являются **RBST**, а также выполняется соотношение P[size(L)=i]=1n,i=1..nP[size(L)=i]=1n,i=1..n.

Из определения следует, что каждый ключ в RBST размера nn может оказаться корнем с вероятностью 1n1n.

Идея RBST состоит в том, что хранимое дерево постоянно является рандомизированным бинарным деревом поиска. Далее подробно будет описана реализация операций над RBST, которая позволит добиться этой цели. Заметим лишь, что хранение RBST в памяти ничем не отличается от хранения обычного

дерева поиска: хранится указатель на корень; в каждой вершине хранятся указатели на её сыновей.

Похожие идеи используются в декартовом дереве, поэтому во многих русскоязычных ресурсах термин **рандомизированное бинарное дерево поиска** используется как синонимическое название декартового дерева и декартового дерева по неявному ключу.

2.2. Описание функций btree.h

Заголовочный файл btree,h содержит структурную реализацию бинарного дерева поиска с рандомизацией, а также все необходимые методы для работы с ним.

2.2.1 Структура Node

Каждый узел двоичного дерева поиска будет содержать ключ key, который по сути управляет всеми процессами (у нас он будет целым), и пару указателей left и right на левое и правое поддеревья. Если какой-либо указатель нулевой, то соответствующее дерево является пустым (т.е. попросту отсутствует). Помимо этого, для реализации рандомизированного дерева потребуется еще одно поле size, в котором будет храниться размер (в вершинах) дерева с корнем в данном узле. Узлы будут представляться структурами:

```
struct node // структура для представления узлов дерева
{ int key;
 int size;
 node* left;
 node* right;
 node(int k) { key = k; left = right = 0; size = 1; }
};
```

2.2.2 Функция поиска

Основное свойство дерева поиска — любой ключ в левом поддереве меньше корневого ключа, а в правом поддереве — больше корневого ключа (для ясности изложения будем считать, что все ключи различны). Это свойство позволяет нам очень просто организовать поиск заданного ключа, перемещаясь от корня вправо или влево в зависимости от значения корневого ключа. Рекурсивный вариант фунции поиска (а мы будем рассматривать только такие) выглядит следующим образом:

```
node* find(node* p, int k) // поиск ключа k в дереве p
{       if( !p ) return 0; // в пустом дереве можно не искать
        if( k == p->key ) return p;
        if( k < p->key ) return find(p→left,k);
        else return find(p->right,k);
}
```

2.2.3 Классическая вставка

В классическом варианте вставка повторяет поиск с тем отличием, что когда мы упираемся в пустой указатель, мы создаем новый узел и подвешиваем его к тому месту, где мы обнаружили тупик. Сначала я не хотел расписывать эту функцию, потому что в рандомизированном случае она работает немного по-другому, но передумал, т.к. хочу зафиксировать пару моментов.

```
node* insert(node* p, int k) // классическая вставка нового узла с ключом k в дерево p {
        if( !p ) return new node(k);
        if( p->key>k ) p->left = insert(p→left,k);
        else p->right = insert(p->right,k);
        fixsize(p);
        return p;
}
```

2.2.4 Функции корректировки размера БДП

Любая функция, которая модифицирует заданное дерево, возвращает указатель на новый корень (неважно, изменился он или нет), а дальше вызывающая функция сама решает, куда «подвешивать» это дерево. Во-вторых, т.к. мы вынуждены хранить размер дерева, то при любом изменении поддеревьев нужно корректировать размер самого дерева. За это отвечает следующая пара функций:

```
int getsize(node* p) // обертка для поля size, работает с пустыми деревьями (t=NULL) {
        if( !p ) return 0;
        return p→size;
 }
void fixsize(node* p) // установление корректного размера дерева {
p->size = getsize(p->left)+getsize(p->right)+1;
}
```

2.2.5 Функции поворота

Необходимой составляющей рандомизации является применение специальной вставки нового ключа, в результате которой новый ключ оказывается в корне дерева. Для реализации вставки в корень нам потребуется функция поворота, которая производит локальное преобразование дерева.

```
node* rotateright(node* p)// правый поворот вокруг узла p {
node* q = p->left;
if( !q ) return p; p->left = q→right;
q->right = p;
q->size = p→size;
fixsize(p); return q; }
```

```
node* rotateleft(node* q) // левый поворот вокруг узла q {
node* p = q->right; if( !p ) return q;
 q->right = p→left;
 p->left = q;
 p->size = q→size;
 fixsize(q);
 return p; }
```

2.2.6 Вставка в корень

Сначала рекурсивно вставляем новый ключ в корень левого или правого поддеревьев (в зависимости от результата сравнения с корневым ключом) и выполняем правый (левый) поворот, который поднимает нужный нам узел в корень дерева.

```
node* insertroot(node* p, int k) // вставка нового узла с ключом k в корень дерева p {
         if( !p ) return new node(k);
         if( k<p->key ) { p->left = insertroot(p→left,k);
                          return rotateright(p); }
        else { p->right = insertroot(p→right,k);
                          return rotateleft(p); } }
```

2.2.7 Рандомизированная вставка

Раз любой ключ (в том числе и тот, который мы сейчас должны вставить в дерево) может оказаться корнем с вероятностью 1/(n+1) (n — размер дерева до вставки), то мы выполняем с указанной вероятностью вставку в корень, а с вероятностью 1-1/(n+1) — рекурсивную вставку в правое или левое поддерево в зависимости от значения ключа в корне:

```
node* insert(node* p, int k) // рандомизированная вставка нового узла с ключом k в дерево p {
         if( !p ) return new node(k);
         if( rand()%(p->size+1)==0 ) return insertroot(p,k);
         if( p->key>k ) p->left = insert(p→left,k);
         else p->right = insert(p→right,k);
         fixsize(p);
        return p;}
```

3. ТЕСТИРОВАНИЕ

3.1. Варианты и ответы

- **1 вариант**

1)Дан дискретный набор узлов БДП: 50 83
Изображение рандомизированного БДП:
50 83

Задание: вставка с рандомизацией узла с весом 63
Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.
Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

2)Дан дискретный набор узлов БДП: 52 75 44 40 6 73 81 14 62 99 9 69 82 1 46 89 5 58
Изображение рандомизированного БДП:
6 62 75 99
81 89
82
73
69
44 52 58
46
9 40
14

5

1

Задание: вставка с рандомизацией узла с весом 0

Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.

Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

3)Дан дискретный набор узлов БДП: 91 12 89 82 4 53

Изображение рандомизированного БДП:

4 82 91

89

12 53

Задание: вставка с рандомизацией узла с весом 68

Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.

Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

- **Ответы**

0.333333

(63(50)(83))

0.666667

(83(50(63)))

0.0526316

(0(6(5(1))(62(44(9(40(14)))(52(46)(58)))(75(73(69))(99(81(89(82))))))))

0.947368

(6(5(1(0)))(62(44(9(40(14)))(52(46)(58)))(75(73(69))(99(81(89(82)))))))

0.142857

(68(4(12(53)))(82(91(89))))

0.857143

(4(82(12(53(68)))(91(89))))

- **2 вариант**

1)Дан дискретный набор узлов БДП: 19 37 29 28 86 68 2 12

Изображение рандомизированного БДП:

28 37 86

68

29

12 19

2

Задание: вставка с рандомизацией узла с весом 96

Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.

Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

2)Дан дискретный набор узлов БДП: 17 68 56 74 34 73 66 41 44 54 69 23 91 36

Изображение рандомизированного БДП:

54 66 73 91

74

69

68
56
44
23 41
34 36
17

Задание: вставка с рандомизацией узла с весом 60
Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.
Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

3)Дан дискретный набор узлов БДП: 90 86 49 100 98 79 64 14 35 25 79 36
Изображение рандомизированного БДП:
90 100
98
86
64 79 79
36 49
25 35
14

Задание: вставка с рандомизацией узла с весом 21
Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.

Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

- **Ответы**

0.111111

(96(28(12(2)(19))(37(29)(86(68)))))

0.888889

(28(12(2)(19))(37(29)(86(68)(96))))

0.0666667

(60(54(44(23(17)(41(34(36)))))(56))(66(73(69(68))(91(74)))))

0.933333

(54(44(23(17)(41(34(36)))))(66(56(60))(73(69(68))(91(74)))))

0.0769231

(21(14)(90(86(64(36(25(35))(49))(79(79))))(100(98))))

0.923077

(90(86(64(14(36(25(21)(35))(49)))(79(79))))(100(98)))

- **3 вариант**

1)Дан дискретный набор узлов БДП: 91 99 83 12

Изображение рандомизированного БДП:

12 99

91

83

Задание: вставка с рандомизацией узла с весом 11

Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.

Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

2)Дан дискретный набор узлов БДП: 5 81 91 82 54 24 79 68

Изображение рандомизированного БДП:

```
 68 82 91
    81
      79
  54
    24
      5
```

Задание: вставка с рандомизацией узла с весом 8

Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.

Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

3)Дан дискретный набор узлов БДП: 31 59 94 87 70 21 91 50 12

Изображение рандомизированного БДП:

```
 94
   12 31 87 91
        59 70
          50
      21
```

Задание: вставка с рандомизацией узла с весом 9

Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.
Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

- **Ответы**

0.2
(11(12(99(91(83)))))
0.8
(12(11)(99(91(83))))
0.111111
(8(5)(68(54(24))(82(81(79))(91))))
0.888889
(5(68(54(24(8)))(82(81(79))(91))))
0.1
(9(94(12(31(21)(87(59(50)(70))(91))))))
0.9
(94(12(9)(31(21)(87(59(50)(70))(91)))))

- **4 вариант**

1)Дан дискретный набор узлов БДП: 74 18
Изображение рандомизированного БДП:
18 74

Задание: вставка с рандомизацией узла с весом 90
Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.
Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

2)Дан дискретный набор узлов БДП: 79 84

Изображение рандомизированного БДП:

84

79

Задание: вставка с рандомизацией узла с весом 8

Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.

Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

3)Дан дискретный набор узлов БДП: 55 27 90 51 79 29 85 90 52 4 59

Изображение рандомизированного БДП:

59 90

79 90

85

4 29 51 52 55

27

Задание: вставка с рандомизацией узла с весом 70

Укажите вероятность, с которой выполнится вставка в корень. Напишите скобочное представление БДП с этом случае.

Укажите вероятность, с которой выполнится рекурсивная вставка в одно из поддерьвев. Напишите скобочное представление БДП с этом случае.

- **Ответы**

0.333333

(90(18(74)))

0.666667

(18(74(90)))

0.333333

(8(84(79)))

0.666667

(84(79(8)))

0.0833333

(70(59(4(29(27)(51(52(55))))))(90(79(90(85)))))

0.916667

(90(59(4(29(27)(51(52(55)))))(79(70)(90(85)))))

3.2. Текущий контроль

- **Smith (1 var)**

  0.333333

  (63(50)(83))

  0.6667

  (83(50(63)))

  (0(6(5(1))(62(44(9(40(14)))(52(46)(58)))(75(73(69))(99(81(89(82))))))))

  0.947368

  (6(5(14(9(40(14)))(52(46)(58)))(75(73(69))(99(81(89(82)))))))

  0.142857

  (68(43)))(82(91(89))))

  0.857143

  (4(82(12(53(68)))(91(89))))

- **Jones (2 var)**

  0.111111

  (96(28(12(2)(19))(37(29)(86(68)))))

  0.888889

  (28(12(2)(19))(37(29)(86(68)(96))))

  0.0666667

  (60(54(44(23(17)(41(34())))(56))(66(73(69(68))(91(74)))))

  0.933333

  (54(44(23(17)(41(34(36))(56(60))(73(69(68))(91(74)))))

  0.0761

  (21(14)(90(86(64(36(25(35))(49))(79(79))))(100(98))))

  0.923077

  (90(86(64(14(36(25(21)(35))(49)))(79(79))))(100(98)))

- **Brown (3 var)**

  0.2

  (11(12(99(91(83)))))

  0.8

  (12(11)(99(91(83))))

  0.111111

  (8(5)(68(54(24))(82(81(79))(91))))

- **Holmes (4 var)**

  0.333333

  (90(18(74)))

  0.666667

  (18(74(90)))

  0.333333

  (8(84(79)))

0.666667

(84(79(8)))

0.0833333

(70(59(4(29(27)(51(52(55))))))(90(79(90(85)))))

0.916667

(90(59(4(29(27)(51(52(55)))))(79(70)(90(85)))))

Результаты ТК:

SMITH:8\12

JONES:9\12

BROWN:6\12

HOLMES:12\12

ЗАКЛЮЧЕНИЕ

В результате курсовой работы был получен опыт по работе рандомизированными бинарными деревьями поиска. На основании этой структуры данных была реализована система Текущего контроля для студентов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://www.wikipedia.org/

2.https://habr.com

ПРИЛОЖЕНИЕ А

```
#INCLUDE "BTREE.H"

USING NAMESPACE STD;

VOID GENERATEARR(VECTOR<INT> &NODES){

    INT N = 1 + RAND()%20;

    FOR (INT I = 0; I < N; I++){

        INT NODE = 1 + RAND()%100;
        NODES.PUSH_BACK(NODE);

    }

}

VOID TASKSGENERATING(STD::OFSTREAM &FILE, STD::OFSTREAM
&ANSW, BINTREESEARCH MYBINTREE, INT &I){

    VECTOR<INT> ARR;
    GENERATEARR(ARR);

    FILE << I << ")ДАН ДИСКРЕТНЫЙ НАБОР УЗЛОВ БДП: ";

    FOR (AUTO K: ARR) {

        FILE << K << ' ';
        MYBINTREE = RANDINSERT(MYBINTREE, K);
```

```
}

FILE << "\NИЗОБРАЖЕНИЕ РАНДОМИЗИРОВАННОГО БДП:\N";
DISPLAY(MYBINTREE, 1, FILE);

INT N = RAND()%100;
FOR( AUTO M: ARR){
        IF( M == N)
            N = RAND()%100;
}

FILE << "\N\NЗАДАНИЕ: ВСТАВКА С РАНДОМИЗАЦИЕЙ УЗЛА С ВЕСОМ " << N;
FLOAT SIZED = ARR.SIZE();
FLOAT CHANCE = 1/ (SIZED + 1);
FILE << "\NУКАЖИТЕ ВЕРОЯТНОСТЬ, С КОТОРОЙ ВЫПОЛНИТСЯ ВСТАВКА В КОРЕНЬ. НАПИШИТЕ СКОБОЧНОЕ ПРЕДСТАВЛЕНИЕ БДП С ЭТОМ СЛУЧАЕ.";
ANSW << CHANCE << ENDL;
MYBINTREE = INSERTROOT(MYBINTREE, N);
PRINTKLP(ANSW, MYBINTREE);

MYBINTREE = REMOVE(MYBINTREE, N);
CHANCE = 1 - CHANCE;
FILE << "\NУКАЖИТЕ ВЕРОЯТНОСТЬ, С КОТОРОЙ ВЫПОЛНИТСЯ РЕКУРСИВНАЯ ВСТАВКА В ОДНО ИЗ ПОДДЕРЬВЕВ. НАПИШИТЕ СКОБОЧНОЕ ПРЕДСТАВЛЕНИЕ БДП С ЭТОМ СЛУЧАЕ.";
ANSW << "\N" << CHANCE << ENDL;
MYBINTREE = SIMPLEINSERT(MYBINTREE, N);
```

```
        PRINTKLP(ANSW, MYBINTREE);
        ANSW << STD::ENDL;

        FILE << "\N\N\N";
        ARR.CLEAR();


}

INT MAIN(INT ARGC, CHAR* ARGV[]){

        SRAND(TIME(0));

        STD::OFSTREAM FILE;
        FILE. OPEN(ARGV[1], STD::IOS::APP);
        IF(!FILE.IS_OPEN()){RETURN 0;}

        STD::OFSTREAM ANSW;
        ANSW. OPEN(ARGV[2],  STD::IOS::APP);
        IF(!ANSW.IS_OPEN()){RETURN 0;}

        FOR (INT I = 1; I < 4; I++){

                BINTREESEARCH MYBINTREE = NULLPTR;
                TASKSGENERATING(FILE, ANSW, MYBINTREE, I);
                DESTROY(MYBINTREE);
        }

        ANSW.CLOSE();
        FILE.CLOSE();
```

```
COUT << "ВВЕДИТЕ СВОЮ ФАМИЛИЮ\N";
STRING SURNAME;
GETLINE(CIN, SURNAME);
STD::OFSTREAM TESTFILE;
TESTFILE. OPEN(SURNAME + ".TXT", STD::IOS::APP);
TESTFILE.CLOSE();

COUT << "\N ЗАДАНИЯ СГЕНЕРИРОВАНЫ. ДЛЯ ПРОХОЖДЕНИЯ ТК ПЕРЕЙДИТЕ В ФАЙЛ " << SURNAME + ".TXT" << ", ВВЕДИТЕ ОТВЕТЫ ПОСТРОЧНО И НАЖМИТЕ \"СОХРАНИТЬ\".\N ДЛЯ ПРОВЕРКИ ВВЕДИТЕ ЛЮБОЙ СИМВОЛ И НАЖМИТЕ ENTER.";
CHAR A;
CIN >> A;
IF (A){

    STD::IFSTREAM TEST(SURNAME + ".TXT");
    STD::IFSTREAM ANS(ARGV[2]);

    STD::STRING S1;
    INT CORRECT = 0;
    WHILE (STD::GETLINE(ANS,S1)){

        STD::STRING S2;
        GETLINE(TEST, S2);
        IF (S1 == S2) CORRECT++;

    }
```

```
            STD::OFSTREAM RESULT;
            RESULT.OPEN("RATING.TXT", STD::IOS::APP);

            RESULT <<  SURNAME << ":" << CORRECT << "\\12\N";

            RESULT.CLOSE();
            TEST.CLOSE();
            ANS.CLOSE();

        }
        RETURN 0;
}
```